JISC Observatory

Observing trends in innovation

Delivering Web to Mobile

Version:	TechWatch Report final version 1.0
Author:	Mark Power
Date:	May 2012

Delivering Web to Mobile, JISC Observatory Techwatch Report, version 1.0, May 2012. http://blog.observatory.jisc.ac.uk/techwatch-reports/delivering-web-to-mobile/

Contents

Introduction 1

State of the Mobile Web 2

UK HEI findings 4

Mobile Web Browsers 5

Responsive Web Design 7

CSS3 media queries 7

Fluid grid layouts 8

Fluid images 9

Mobile First 10

Progressive Enhancement 10

Server-side Device Detection 11

Dedicated Mobile Site? 12

Mobile Web Apps 12

HTML5 14

Device APIs 16

HTML5 Frameworks 17

HTML5 Mobile Boilerplate 17

Jo 17

jQuery Mobile 18

“Hybrid Apps” 18

PhoneGap 19

Appcelerator Titanium 20

Corona 20

Introduction

This report is intended to help staff of UK education institutions, involved in the development of content, gain an understanding of the emerging approaches to delivering services and content for mobile devices using the Web.

The use of mobile devices for the consumption and use of Web content and services has grown steadily over the last few years and continues to do so, with analysts predicting that mobile will soon exceed the traditional desktop PC as the most common means users interact with the Web and other Internet services.

This report looks at the growth of mobile, the state of the Web and gives an overview of approaches to delivering content and services optimised for the mobile context. This includes approaches to Web design for responsive sites, leveraging access to device functions and capabilities and the use of Web technologies to build mobile applications.

Whilst it is recognised that accessibility is a key issue in the creation and delivery of web content, given the scope of the topic this is an area that is not covered in this particular report. It is suggested staff wishing to investigate issues around web accessibility visit JISC TechDis for advice and guidance at http://www.jisctechdis.ac.uk/

> “There's no need to declare this "the year of mobile." If anything, last year was the year of mobile in terms of the growth in both mobile usage and the availability of mobile sites and apps. Now, however, it's time to redesign your mobile site, because your existing version is probably far below users' growing expectations for user experience quality.”
>
> Jakob Nielsen (September 2011)

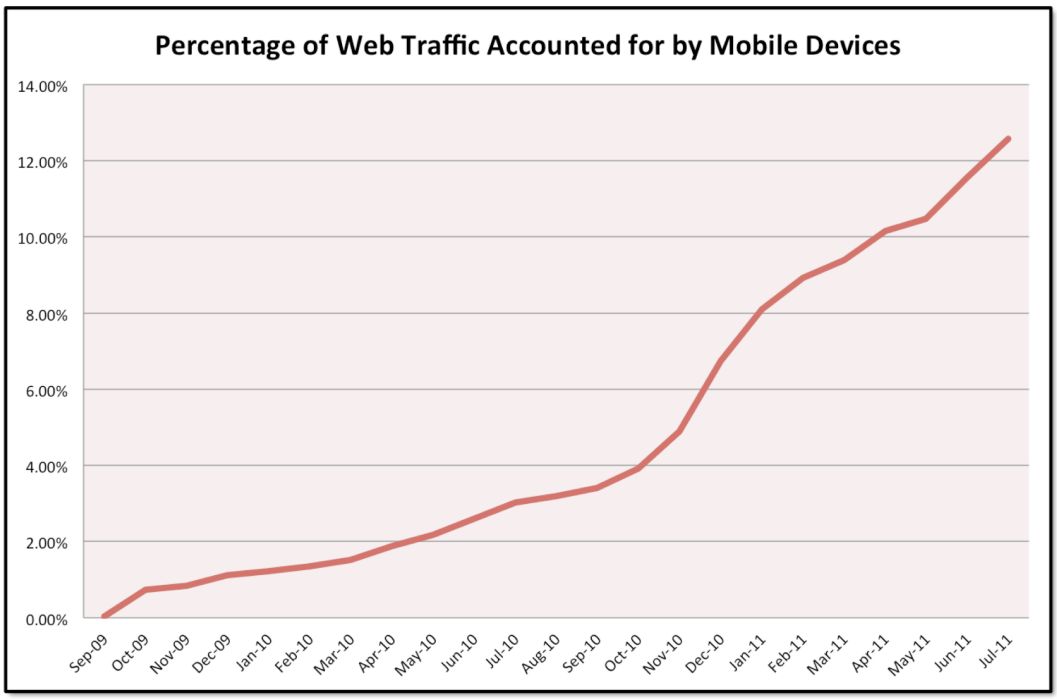
http://www.useit.com/alertbox/mobile-usability.html

State of the Mobile Web

With increased sales in smartphones and tablets, faster data connections, improving hardware and the explosion of mobile apps, the use of the Internet on mobile devices is rapidly growing.

Smartphones are getting larger and faster with HD displays and quad-core processors, outstripping many of the desktop PCs used in education, business and the home just a decade ago.

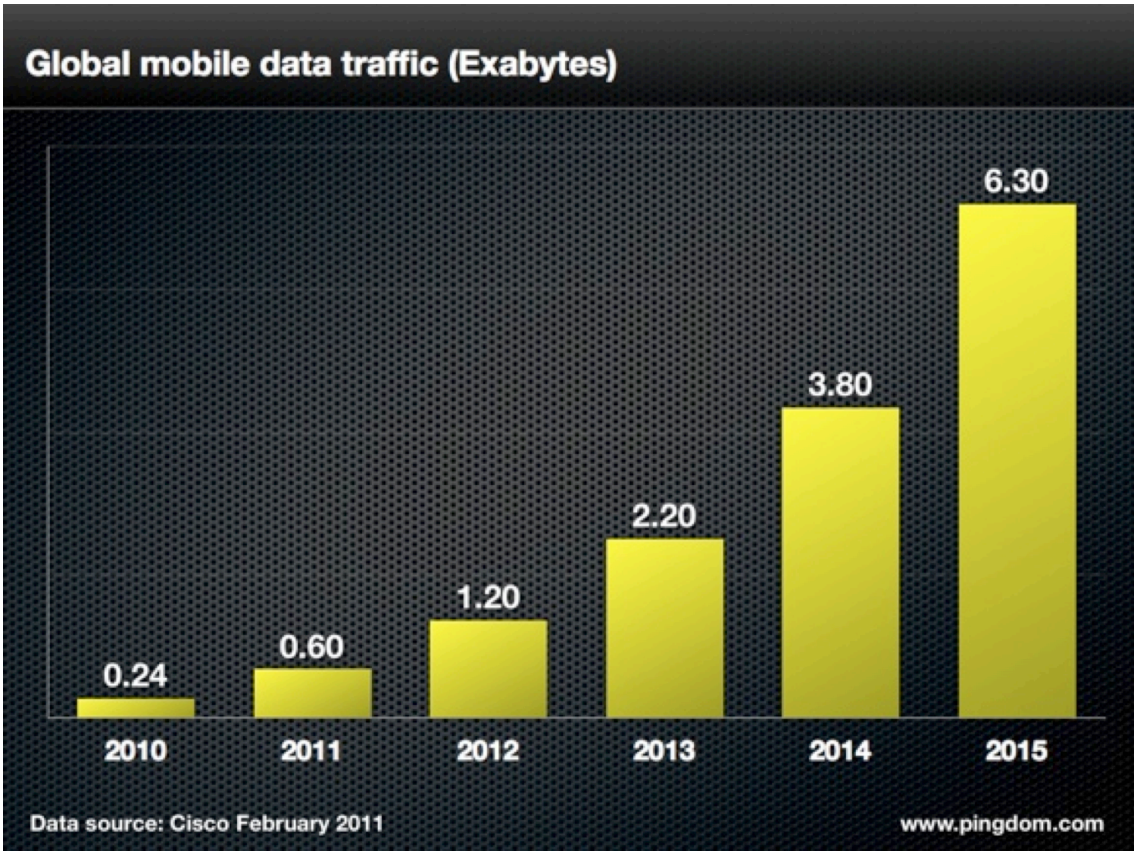
A 2011 study by the digital marketing agency, Tecmark, suggests that mobile now accounts for 12.59% of UK Web traffic and notes that the consumption of the Web on mobile devices has increased steadily, month-on-month, since the end of 2009.



Figures from other sources back this up, with the website GlobalStats showing that mobile now accounts for around 6.5% of global web traffic coming from mobile devices (http://gs.statcounter.com/#mobile_vs_desktop-ww-yearly-2011-2011-bar).

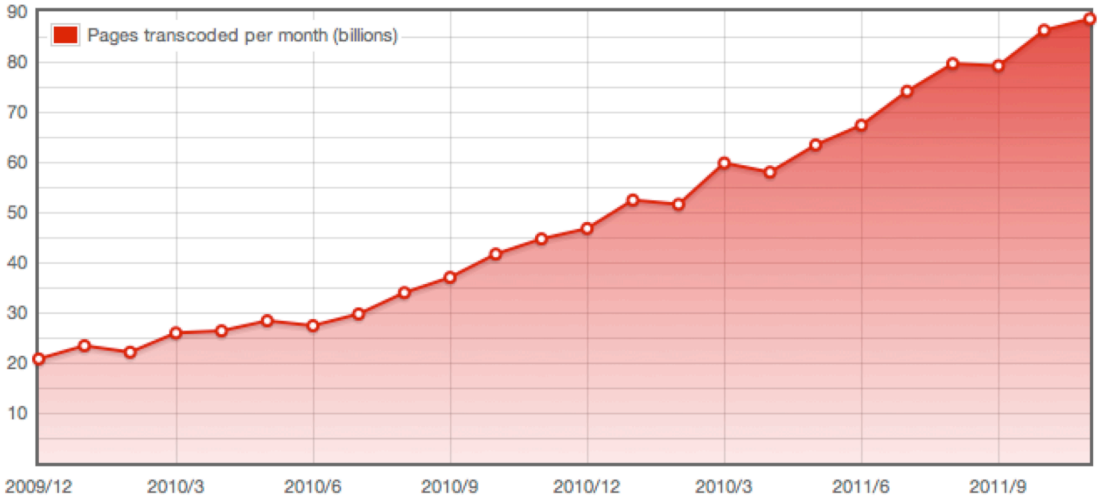
While the figures for mobile use of the web are still relatively small compared to traditional desktop access, it cannot be ignored that the use of the web on mobile is steadily growing and will continue to do so, with several tech analysts predicting that by 2015 the use of the web on

mobile devices will outstrip that of desktop traffic (see ITU prediction at http://www.itu.int/newsroom/press_releases/2010/06.html)



Every month the browser software company, Opera, conducts a definitive analysis of the key trends affecting the mobile web and publishes these as their State of the Mobile Web reports (http://www.opera.com/smw/). These are useful reports as Opera not only provides a browser for smartphones but also feature phones, which still account for the lion's share of mobile phone usage.

Again, their findings reflect the huge ongoing growth in web traffic to mobile devices (theirs being specific to their own ubiquitous Opera Mini mobile browser).



The November 2011 State of the Mobile Web report contained two notable figures:

- An increase of 84.2% in unique users of Opera Mini mobile web browser from Sept2010
- An increase of 114.1% in page views on OM from Sept 2010

UK HEI findings

Surveys of UK universities conducted through the Web Information Manager JISCMail list similarly reflected a growth in web activity on mobile devices, statistics showing that visits to university websites from mobile browsers increasing (in some cases) as much as 200% between November 2010 - 2011.

Again, it should be stated that the actual percentage of overall web visits from mobile, compared to desktop traffic, is still quite small – around 2% in this survey – but it is apparent that access through mobile is growing across the sector just as it is across general web usage and this means that UK F/HE institutions need to be looking now at how they address this when it comes to delivering web content and services to their users (both students and staff).

Mobile Web Browsers

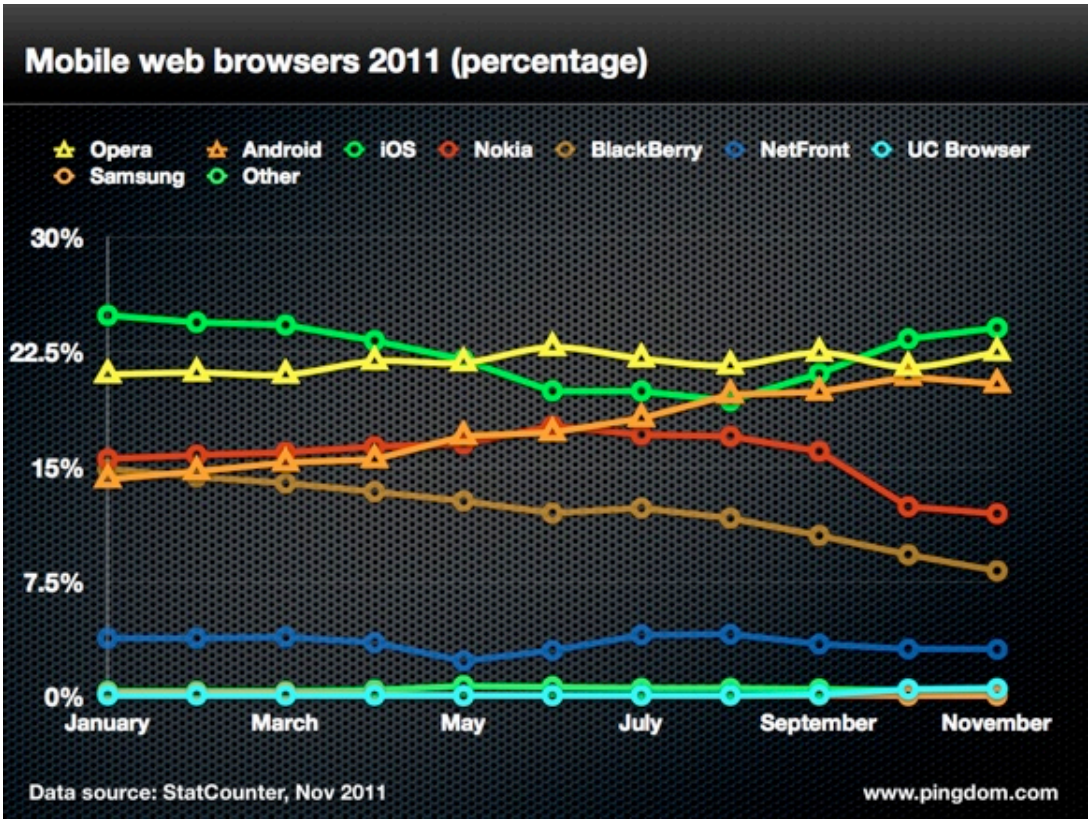
The web on mobile used to mean heavily trimmed-down pages accessed over WAP (Wireless Application Protocol) and other slow connections, consisting of mainly text-based content delivered to phones with small screens, limited colours and generally very basic web capability.

Given the limitations in screen size and device capability it's unsurprising that the experience of using the web was a fairly unsatisfactory and frustrating one.

The arrival of the first iPhone from Apple, in 2007, saw the game change when it comes to viewing and using the web on mobile devices. The iPhone gave us a smartphone that could present fully rendered web pages on which users could zoom in, pan around, view full colour images and watch videos (ignoring the lack of Flash support).

The ever increasing power of today's mobile devices has, naturally, led to an increase in the capabilities of their web browsers.

These are not just on high-end smartphones either. Even lower-end feature phones are beginning to include rich web browser features. Given this increase in capability in the mobile browser market there are new techniques available to web developers to enable them to cater more easily for this vast array of browsers, devices and screen sizes.



Given these numbers and the state of the technology it's looking clear that institutions should be considering their options when it comes to delivering their content and services to mobile. How best to approach catering for the variety of devices that are classed under the term "mobile" – feature phones, smartphones, tablets? Is the aim to deliver a simple web experience, flexible enough to work sufficiently well across a variety of screen sizes? Is it to deliver a fully designed-for-mobile experience that runs across all devices, not just smartphones and tablets? Does it make more sense to have a separate, dedicated mobile website? Is there a need for a context-aware service in app form?

The W3C offer some Mobile Web Best Practices at http://www.w3.org/TR/mobile-bp/

Responsive Web Design

This approach was popularised by the web designer Ethan Marcotte in an article for A List Apart (http://www.alistapart.com/articles/responsive-web-design/) in May 2010. Ethan has since published his book: Responsive Web Design: http://www.abookapart.com/products/responsive-web-design

Responsive web design is not about building “mobile websites”. Marcotte outlined a set of design principles that allow a website to be flexible enough to work well at varying resolutions, that will fluidly scale and fit to suit the screen it is being viewed on; from phone, to tablet, to desktop and further.

Responsive web design uses three primary techniques:

- **CSS3 media queries**
  Allows for delivery of tailored styles to suit the browser environment
- **Fluid-grid layouts**
  Enables the underlying page grid to scale nicely, using relative proportions rather than fixed pixel dimensions
- **Fluid images and other media**
  Enables images (and video) to scale effectively within the grid

CSS3 media queries

Depending on certain attributes of the user’s browser; such as screen size, orientation (portrait v. landscape) or aspect ratio, different CSS (Cascading Style Sheets) can be applied that optimises the presentation of your website to fit and flow the browser on which the site is being viewed. The beauty of this approach is that, in applying it, your websites can be optimised for small screen feature phones, larger screen smartphones, tablets, traditional ‘desktop’ browser views and upward for huge screens such as televisions.

Since the specification of CSS 2.1, style sheets have had some measure of capability when it comes to device recognition through the use of media types. For example:

```
<link rel="stylesheet" type="text/css" href="print.css"
  media="print" />
```

has allowed us to apply separate rules to the styling of our pages, formatted nicely for print. However, while the specification included a collection of media types, targeted at identifying device classes, these often went ignored and unused by browsers and devices.

With CSS3, the W3C built on and improved media types with media features and media queries. These not only make it possible to inspect the device that the content is being delivered to but also the actual physical characteristics of the device.

Using media queries web developers can easily ask the browser about its features, such as width, height, aspect-ratio and orientation. So by applying a simple line of code such as:

```
<link rel="stylesheet" type="text/css" media="screen and (max-
width: 480px)" href="480.css" />
```

we can then deliver a tailored stylesheet (480.css) to devices that have a maximum width of 480px.

Media queries aren't limited to use in HTML links to stylesheets either. They can be incorporated into the CSS itself as a media rule.

```
@media screen and (max-width:480px) { /* CSS Rules */ }
```

Using this query the stated CSS rules are only applied if the media query evaluates to TRUE. This way font sizes, colours, navigation and column layouts can be tailored to suit the dimensions of the screen.

For the full list of W3C media features, see: http://www.w3.org/TR/css3-mediaqueries

Fluid grid layouts

Many websites have traditionally been built using fixed-width layouts. In the late 90s desktop monitors were (relatively) low resolution screens and web designers tended to agree on designing to fit the 800 x 600 pixels view used by most web users. However, as screens improved and resolutions went up designers started to scale up their designs and generally ended up providing for the lowest common resolution of 1024px (width).

However, fixed-width pixel dimensions are inflexible and can't adapt to different resolutions. Instead, relative proportions using percentage values allow the layout to flow more naturally with the resolution and maintain their intended shape.

Fluid images

In creating flexible designs that use percentages instead of fixed pixel widths you enable your content to fit the screen more naturally. The same rule then needs to be applied to images (and video). Using a 600px wide image in a column that then scales down below that size is going to cause problems so best practice in this technique is to set a percentage width to media too. These then scale naturally within their HTML containers.

For some excellent examples of sites using a responsive design approach, visit http://mediaqueri.es The site provides a clear view of how website creators are using these techniques to deliver responsive sites.

Example of a responsive web design from http://mediaqueri.es/

Mobile First

Taking the responsive web design approach further, several people have started to champion the sense in designing sites first for mobile, then scaling up to suit larger resolutions using progressive enhancement.

One of the problems with responsive web design – the fluid images approach specifically – is that the mobile browser still has to download full-size images, which then get scaled to fit. This can be problematic from the point of view of file sizes and connection speeds. Designing for mobile first means that such media can be delivered to be optimised more efficiently for mobile. Best practice in this approach is to initially serve mobile-friendly images to all devices and then, dependent on browser, specify desktop-sized images to replace them.

Another benefit of the mobile first approach is that it can focus the content provider on the essentials of what they want to deliver to the mobile user and cut out unnecessary elements.

However, this approach does mean starting from scratch. If the web build is a new project, starting from the ground up, then the mobile first approach has its benefits.

One of the key champions of this approach is Luke Wroblewski of Ideation + Design. Luke's site contains articles and presentations on the mobile web and the mobile first approach – http://www.lukew.com/

Progressive Enhancement

Progressive enhancement has been around in web development for a number of years now, used to adapt sites on browsers according to their capabilities. The idea behind this technique is to deliver a single base-page to every device with JavaScript enhancement. If the device is very basic and doesn't run JavaScript then the user still gets the usable base-content. If the browser is more capable the JavaScript runs, adding enhancements in functionality progressively to meet the browser's level of support.

The strength of progressive enhancement is being able to cater for a much wider spectrum of devices by including low level feature phones. However, more functionality means more code which in turn means more time to load and execute. So even if a device is capable of supporting JavaScript enhancements, developers should bear in mind what the knock-on effects of these are to the user.

Server-side Device Detection

This approach relies on the use of a device detection library/database installed on the web server. This library contains detailed information about the mobile device and on being queried from an API in the page, returns the attributes specific to the device.

Given this data the developer can adapt content based on the device's features and capabilities. This is more powerful than the simple media query technique of responsive web design as such a library contains a large amount of data on not only the resolution of the device but its platform, browser and hardware features.

This does mean that such a library needs to be constantly up-to-date and WURFL does just that. **WURFL** (Wireless Universal Resource FiLe) is an open source device database that tracks and maintains details about mobile devices.

For an example of the information WURFL can provide, visit http://www.tera-wurfl.com/explore on your mobile device and select 'See my capabilities'. The image on the right shows the product info from the iPhone 4.

There are currently over 500 capabilities across thousands of devices that WURFL tracks and the data is maintained and regularly updated. This is available from wurfl.sourceforge.net This is an XML configuration file (containing the device data) and a set of programming APIs to access the data.

It is important to note though that there are licensing issues when it comes to using WURFL and projects have had to fork it or drop it completely and look to alternatives.

One alternative is the **OpenDDR** (Device Description Repository). OpenDDR offers device libraries, maintained by the open source community and contributed to by major device manufacturers, along with W3C DDR Simple APIs. These are license free, open standards compliant and available from http://openddr.org/

Another option taking shape is the **Apache DeviceMap project**. At time of writing the project is still in its early development stages and propose to initially focus on basic device features as used in HTML5 websites and applications. Given the standing of Apache and its well established open source community, this project is worth keeping track of. The DeviceMap project can be found at http://incubator.apache.org/projects/devicemap.html

Dedicated Mobile Site?

While responsive web design is a helpful, broad approach to creating websites that are flexible and adapt to differences in resolutions across mobile devices (and upward), it is still only that. It is only a set of design principles and techniques that allows your web content to adapt to the screen.

Another option for institutions is creating dedicated mobile sites. This can be a bigger job of course because it means having to essentially build two versions of the same site; with different stylesheets, templates and maybe even different content. The benefit though is that the dedicated site can be more effectively optimised for mobile – using made-for-purpose images and other media, meaning less content to load.

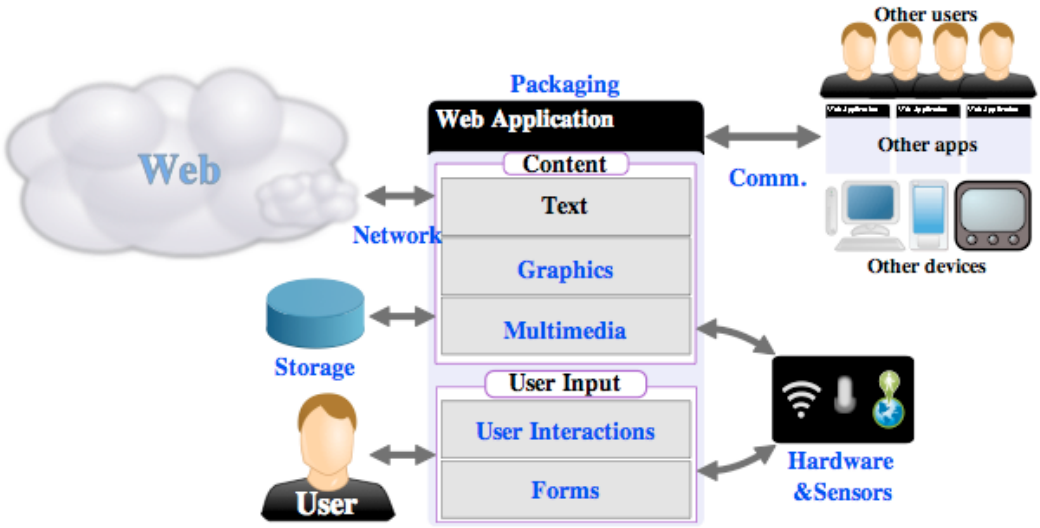
Mobile Web Apps

With the rise of the smartphone the mobile scene has also witnessed the now dominant paradigm in consumption of content and usage of tailor-made services on mobile devices with applications. Search, shopping, news readers, games and social networking on mobile is catered for by a huge number of native apps that are bought through dedicated app stores and installed directly onto the users' devices.

However, while native mobile applications can be very powerful and feature rich offerings that have access to all the device capabilities and run seamlessly using the processing power of the devices, developing native apps can be a costly and resource-heavy approach due to each ecosystem using different native code development and separate deployment to each platform. This means if you want as many users as possible to be able to use your app then you have to build multiple versions to cover the variety of platforms.

For institutions delivering services to mobile the challenge of catering to a market that is so fragmented and constantly shifting as market shares change and new players in mobile rise – such as Microsoft's re-emergence with Windows Phone – and fall, such as Nokia's Symbian is a huge one. Fortunately the mobile web is no longer limited to simply delivering 'plain' web content optimised for smaller screens but is now making serious in-roads into being a viable alternative for building cross-platform mobile applications that need no dedicated app store, no 3rd party approval processes and can be developed using existing (and improving) open web technologies, with development skills that institutions already have available to them in their existing web designers and developers.

While there is still a gap between the capabilities of mobile web apps and their native counterparts, work continues on open web technologies that looks to close this gap and make the web a powerful and viable option for creating feature-rich applications.



The Web as an application development platform (source: *http://www.w3.org/2011/02/mobile-web-app-state.html*)

For a focused overview of the technologies in mobile web apps, see the JISC CETIS Briefing Paper available at http://wiki.cetis.ac.uk/images/7/76/Mobile_Web_Apps.pdf

For an in-depth look at the state of mobile web standards, visit the W3C site at http://www.w3.org/2011/02/mobile-web-app-state.html

This summarises the various technologies developed in W3C that increase the power of the web as an application platform and how they apply – more specifically – to the mobile context. The W3C website covers the features that these technologies add to the Web platform under the following categories:

Graphics	Multimedia	Forms	User Interactions	Data Storage
Sensors & hardware integration	Network	Communication	Packaging	Performance & Optimisation

HTML5

Few can have escaped noticing the growing buzz behind HTML5 and the promise it holds of 'native-like' applications that can run across any device, on any operating system, using the web browser.

In actual fact, the promise of context-aware and richer mobile web apps requires the combination of several web technologies, not just HTML5 alone.

The HTML5 standard alone introduces new semantic elements such as <nav>, <header> and <article>. However, when it comes to "apps" the most powerful things it gives us is browser-native media through new elements for <audio> and <video> – thereby removing the need for 3rd party plugins like Flash, that lack support on Apple's platform, iOS – and allows for more interactivity through new JavaScript APIs such as geolocation and offline storage.

However, much of the time the term "HTML5" is used as the name for the combination of HTML5 itself, with new presentation tricks delivered via CSS3 and behaviours and functionality through JavaScript.

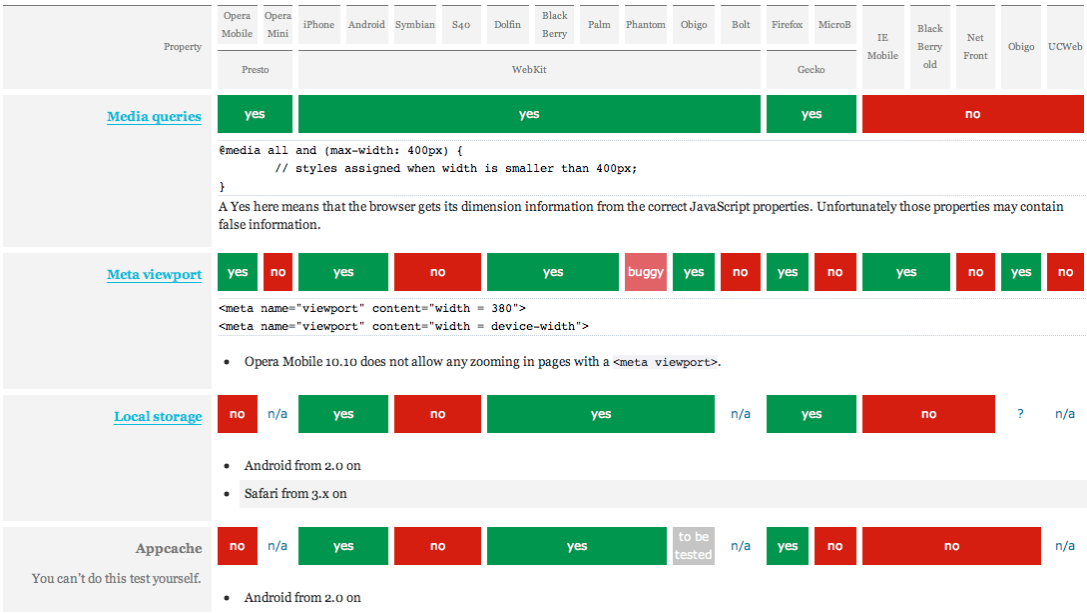
Some of the key features that the combination of these web technologies immediately offer are outlined in the table below.

Plugin-free media	**Local storage**
With HTML5's introduction of the new `<video>` and `<audio>` elements, web developers can now include media within their pages without the need for embedding it in a plugin like Flash. Given the high profile case of Apple refusing to support Flash on their massively popular iOS devices, this is something of a Godsend. HTML5 also gives developers a simple means by which to create interface controls.	The HTML 5 specification contains a standard for local storage that is implemented by a wide variety of browsers. Using the localStorage API You can create applications that store their data locally on the user's phone rather than on your servers. This can be used to enable applications that use dynamic data such as calendars to also be used offline, or to support personalization of the app by users without them needing to log in or have an account on your site.
Offline applications	**Geolocation**
Giving your apps some level of offline capability can bring it closer to the native experience as your key interface features – buttons, images, styles, scripts, etc – can all still work even if the user has a	W3C's Geolocation API is a simple JavaScript API that when plugged into your app can enhance the user's interaction with your service by pinpointing their exact position using the GPS sensors built into today's devices. Supported by many of today's

poor internet connection. HTML5 enables developers to specify which files should be stored locally on the user's device. This saves your app from ending up being a blank page if the user doesn't have a connection, thus improving the user experience.	Mobile WebKit browsers on all the main platforms. Google Maps uses this for their mobile web app.
Multi-threaded JavaScript	**Easy form handling**
HTML5's Web Workers specification provides applications with the ability to use scripts that run in the background without interacting with users. These can be used for long-running tasks or functions that require a lot of computation. For example, a complex calculation for something like bursary allowances could be run using a Web Worker in an offline client rather than using a web service. Another use is in scientific applications, again enabling client-side computation that is normally considered too "heavyweight" for the browser to perform. As well as computation, Workers can be used for other long- running activities such as monitoring the results of a request that takes a long time such as federated search results, or to continuously monitor an activity stream for relevant items.	Your mobile users will often have to key in information on your app. HTML5 brings new form types that are recognised by the browsers and formatted accordingly, presenting the user with the keyboard they need, no longer needing lengthy JavaScripts... just declared straight in the HTML

All modern mobile web browsers offer rich support for HTML5 and CSS3 but there are gaps and there are, of course, older browsers that may lack some of the capabilities or handle things differently.

For information on various aspects of mobile web development, including browser comparisons and compatibility tables visit www.quirksmode.org/mobile, run by the mobile platform strategist and web specialist, Peter-Paul Koch.



Snippet of mobile browser compatibility table at www.quirksmode.org/mobile

Device APIs

Device APIs are client-side APIs – written in JavaScript – that enable the development of web applications that can interact with the device hardware, for example; the camera, GPS, compass and accelerometer, as well as hooking up your web app with other functions like the calendar, the messaging system and address book to create more context-aware web applications.

These APIs are of key importance when looking to develop mobile web apps, rather than simply looking to optimise plain web content to fit mobile screens. Currently, device APIs are undergoing the standardisation process through two main groups – the W3C's Device APIs Working Group (DAP) and the Wholesale Applications Community (WAC).

While some are already robust and widely supported across devices – Geolocation being a prime example – others are lacking widespread support at time of writing but continue to develop and gain traction.

The latest state of these specifications can be found at:

W3C DAP: http://www.w3.org/2009/dap

WAC: http://specs.wacapps.net

http://mobilehtml5.org/ is a table of up-to-date information on the current support for W3C APIs on the most common and popular mobile web browsers.

HTML5 Frameworks

HTML5 Mobile Boilerplate

http://html5boilerplate.com/mobile

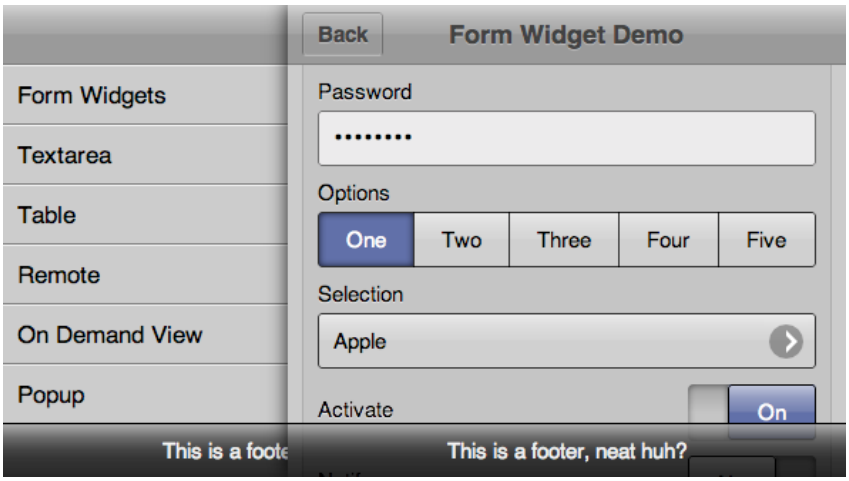
The Mobile Boilerplate is a build tool template that can be built on and customised, allowing developers to create rich and standards-conformant mobile web apps. Boilerplate is not a framework, nor does it prescribe any philosophy of development, it's just got some tricks to get your project off the ground quickly and right-footed.

It quickly enables developers to create mobile web apps and sites that are cross-platform compatible with all the major platforms (Android, iOS, Blackberry, Symbian) and the template has built-in cross-platform viewport optimisation. The template also includes HTML5 offline caching and mobile optimised default CSS.

Jo

http://joapp.com

Specifically for mobile web apps, Jo is a lightweight JavaScript framework for HTML5 allowing easy creation of native-like widgets and UI controls. Jo is open source and free to use.



Screenshot of Jo's Widget Demo

The website states its philosophy as such:

If you want to jam an existing web page into an application framework, Jo probably isn't for you. Jo is designed to create applications. While it will play nicely with a mixture of new and old web development techniques, it uses HTML5 as a development stack and does not require direct DOM manipulation.

jQuery Mobile

http://jquerymobile.com

jQuery Mobile describes itself as a touch-optimised web framework for smartphones and tablets. It provides an HTML5-based user interface system, built on the jQuery and jQuery UI libraries.

Source: www.jquerymobile.com

The use of jQuery Mobile is for separate mobile websites or web apps that can then be natively-wrapped as a hybrid app (see later section). The UI elements are basic and fairly generic but can be styled through the CSS. A useful tool to assist in this is ThemeRoller for Mobile (http://jquerymobile.com/themeroller)

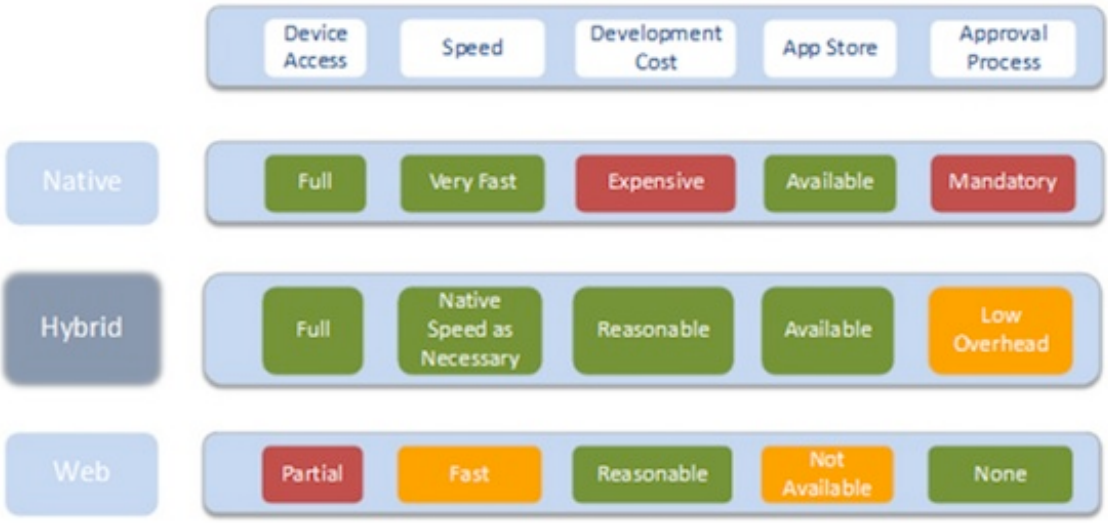
“Hybrid Apps”

For much of 2011, one of the biggest debates in the mobile app scene was “native v. web”. While native apps do deliver the most optimised solution for devices – with their natively run code powered by the device and full access to hardware and device features – the challenge in developing apps across all platforms to cater for the massively fragmented market of users is a very costly and resource heavy obstacle for institutions, requiring the use of different coding languages, frameworks and app store deployments.

While web apps are a viable solution to creating cross-platform mobile applications, there may be reasons developers wish to provide a native app. Wider access to device functionality, graphics heavy interfaces or perhaps from a marketing perspective of having a presence in the platform app store all balance out overall requirements of what it is the institution is wanting to provide for their mobile users.

Fortunately there is a third option that combines the use of web technologies with native capabilities and deployment – Hybrid Apps. These are native apps that are built using web technologies that are then 'packaged' in a native shell, integrating access to device APIs and app stores. Hybrid apps enable developers to bridge the gap between web and native, building an application once using web technologies and then extending and wrapping this core application in multiple formats for deployment across multiple platforms.

The table below highlights the key pros and cons of these different approaches to app development and hybrid apps are clearly an interesting option, giving developers some of the best of both worlds of web and native.



Source: Worklight – http://www.worklight.com/resources/webinars-and-tools/native-web-hybrid-mobile-app-development

Building hybrid apps does require the use of a development framework, of which there are several to choose from, including:

PhoneGap

http://www.phonegap.com

PhoneGap is a popular choice amongst developers of hybrid apps, embedding web apps in a native shell and enabling developers to utilise native device APIs like the camera and contact list using JavaScript. It is a standards-based, open source development framework, free to download, with community-built development tools and plugins.

PhoneGap also offers their PhoneGap:Build service. This cloud-based service enables developers to upload their web apps and get back app-store ready native apps for Apple iOS, Android, Palm, Symbian, Blackberry and others.

Appcelerator Titanium

http://www.appcelerator.com

Another popular solution, Titanium offers their SDK and basic mobile APIs for free, with add-on subscription packages for more advanced APIs and tools for Enterprise.

Corona

http://www.anscamobile.com/corona

Corona is a subscription solution whose SDK enables developers to build a single app that can then be deployed on iOS and Android.